MEMBANGUN SISTEM ANALISIS MALWARE PADA APLIKASI ANDROID DENGAN METODE REVERSE ENGINEERING MENGGUNAKAN REMNUX

Yudha Aprianto Utomo[1], Setia Juli Irzal Ismail S.T., M.T.[2], Tafta Zani S.T., M.T.[3]

1, 2, 3, Prodi D3 Teknik Komputer, Fakultas Ilmu Terapan, Universitas Telkom
[1]dhapriantoutomo@gmail.com, [2] jul@tass.telkomuniversity.ac.id,
[3]tafta@tass.telkomuniversity.ac.id

Abstrak

Pada generasi milenial seperti sekarang sudah banyak terjadi kejahatan siber, dan itu semakin berkembang dengan pesat. Salah satu kejahatan siber yang menghawatirkan adalah dengan menggunakan *malware*. Saat ini *malware* sudah mulai masuk ke *smartphone* dengan perantara aplikasi. Untuk mengantisipasi masuknya *malware* kedalam *smartphone*, perlu adanya proses analisis dengan metode yang tepat dan mudah digunakan. Salah satu metode yang digunakan adalah dengan meggunakan metode *reverse engineering* atau rekayasa balik yaitu metode untuk mencari suatu informasi adanya malware yang disembunyikan. Untuk mendukung metode *reverse engineering* terdapat sistem operasi yang bernama Remnux yang merupakan salah satu distro dari linux. Di dalam sistem operasi Remnux terdapat tools yang dapat menganalisis malware dalam bentuk apk, xml dan dex yaitu *androguard*. Dari hasil pengujian yang telah dilakukan bahwa sistem analisis aplikasi android ini dapat melakukan analisa terhadap izin aplikasi, nama paket, aktivitas utama dan kode program dari setiap *class*. Terdapat 3 sampel aplikasi versi palsu yang dianalisa dan selanjutnya dibandingkan dengan 3 sampel aplikasi yang didapatkan dari Google Play Store. Hasil akhir yang didapat dari pengujian ini adalah terdeteksinya *malware* pada kode program dari 3 aplikasi versi palsu dan mengetahui cara kerja *malware* tersebut.

Kata kunci: ***Malware, Remnux, Analisis, Reverse Engineering***

Abstract

Millenials like now have a lot of cyber crime, and it's growing rapidly. One of the cyber crimes that concerns is to use malware. Currently malware has begun to enter into smartphones with an intermediary of applications. To anticipate the entry of malware into the smartphone, the need for an analysis process with the right method and easy to use. One of the methods used is by using reverse engineering method that is to search for information about hidden malware. To support reverse engineering methods there is an operating system called remnux which is one of the distro of linux. Inside the operating system remnux there are tools that can analyze malware in the form of apk, xml and dex. namely androguard. From the results of testing that has been done that the Android application analysis system can analyze the application permissions, package names, main activities and program codes of each class. There are 3 fake version application samples that were analyzed and then compared with 3 application samples obtained from the Google Play Store. The final result obtained from this test is the detection of malware on the program code from 3 fake version applications and knowing how the malware works.

Keywords: ***Malware, Remnux, Analysis, Reverse Engineering***

1. PENDAHULUAN

1.1 Latar Belakang

Malware atau dalam bahasa Indonesia perangkat lunak berbahaya adalah perangkat lunak yang diciptakan untuk menyusup atau merusak sistem komputer, server atau jejaring komputer tanpa izin termaklum dari pemilik. Istilah ini lazim digunakan oleh pakar komputer untuk mengartikan berbagai

macam aplikasi atau kode yang berniat jahat atau destruktif. Sistem operasi yang terdapat dalam *smartphone* yaitu Android juga dapat terserang *malware*. *Malware* telah dirancang secanggih mungkin untuk membuat celah pada sistem keamanan pada suatu komputer. Berbagai cara proteksi keamanan tidak sepenuhnya dapat menjadi jaminan sistem aman dari serangan *malware* karena setiap *malware* diberikan teknologi pertahanan tersendiri untuk melindungi dirinya dari segala ancaman.

Ada cara yang lebih rinci untuk mengetahui cara kerja suatu *malware* dalam suatu sistem, yaitu dengan menggunakan metode *Reverse Engineering. Reverse engineering* yaitu suatu metode yang digunakan untuk menganalisis sebuah *malware* dengan cara menganalisa kode program dari sebuah *malware* yang akan diteliti. Sistem operasi yang cocok untuk menganalisis suatu *malware* salah satunya adalah Remnux. Di dalam Remnux terdapat *tools* yang dapat digunakan dalam pengerjaan proyek akhir ini. Tentu saja hal itu lebih mudah apabila menggunakan metode *reverse engineering*, sehingga dapat dengan jelas mengetahui suatu *malware* dapat membuat celah dalam sistem dan juga dapat melihat komponen dari *malware* itu sendiri.

Sebelumnya telah ada penelitian tentang analisis *malware* yang menggunakan Remnux dengan judul *"Analisis Malware Menggunakan Metode Reverse Engineering Pada Remnux"* yang di buat oleh Muhammad Rizal Zulfikar [1], penelitian dilakukan adalah dengan menganalisis *website* dan juga *file* yang berformat EXE dan PDF. Pada proyek akhir ini akan menganalisis *file* yang berformat APK, dimana itu adalah format berkas sebuah aplikasi yang ada pada sistem operasi Android.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang diuraikan sebelumnya, maka dapat disimpulkan masalah yang terjadi sebagai berikut :

1. Alat yang tersedia untuk membantu analisis *malware* android membutuhkan banyak *tools*.
2. Bagaimana mengetahui adanya kode program yang mengandung *malware* pada sebuah aplikasi?

1.3 Tujuan

Adapun tujuan dari proyek akhir ini adalah sebagai berikut :

1. Membangun aplikasi yang dapat membantu menganalisa *malware* android.
2. Mendeteksi keberadaan *malware* android yang ada pada kode program sebuah aplikasi.

1.4 Batasan Masalah

Batasan masalah dapat berisi:

1. Menggunakan sistem operasi Remnux versi 6,
2. Sistem operasi dijalankan dengan mesin virtual,
3. Hanya menganalisis *malware* yang terdapat pada sistem operasi Android,
4. Tidak ada perubahan pada sampel *file* APK,
5. Tidak menangani lebih jauh tentang pencegahan dan penanganan terhadap *malware*,
6. Tidak membahas tentang bahasa *Assembly* dan *Disassambly*.

2. TINJAUAN PUSTAKA

2.1 Penelitian Sebelumnya

Pada penelitian sebelumnya, Rahmat Novrianda, Yesi Novaria Kunang dan P.H. Shaksono yang berjudul "ANALISIS FORENSIK MALWARE

PADA PLATFORM ANDROID" menganalisa 3 sampel *malware* berformat APK menggunakan 3 *tools* yaitu, Winrar, dex2jar dan JD-GUI yang selanjutnya langsung menganalisa kode program aplikasi [2].

Pada penelitian sebelumnya, Muhammad Rizal Zulfikar yang berjudul "ANALISIS MALWARE MENGGUNAKAN METODE REVERSE ENGINEERING PADA REMNUX" menggunakan sistem operasi Remnux yang di impor ke sebuah mesin virtual yang selanjutnya melakukan analisa terhadap situs web serta *file* yang berformat EXE dan PDF [1].

2.2 Teori

2.2.1 Malware

Malware atau dalam bahasa Indonesia disebut perangkat lunak berbahaya, merupakan perangkat lunak yang diciptakan untuk menyusup atau merusak sistem komputer, server atau jejaring komputer tanpa izin dari pemilik. Bukan hanya menyerang komputer, saat ini *malware* juga sudah masuk menyerang ke *smartphone*. *Malware* banyak ditemukan menyerang *smartphone* dengan sistem operasi Android dikarenakan populernya sistem aplikasi ini di mata masyarakat. Pada bulan Mei 2013, Google melaporkan 900 juta perangkat Android telah aktif di seluruh dunia dan 48 miliar aplikasi telah dipasang dari Google Play [3] [4]. Karena banyaknya pengguna perangkat Android muncullah aplikasi-aplikasi yang mengandung *malware* dengan tujuan mencari keuntungan dengan mencuri data dari pengguna, dan itu dapat kita temukan pada toko resmi aplikasi Google Play. Data statistik penyebaran *malware* di Indonesia pada tahun 2014 menunjukan bahwa ada 4 jenis malware yang menjadi ancaman [5].

Gambar 2. 1 Statistik Penyebaran Malware di Indonesia

Pada gambar 2.1 menunjukan bahwa penyebaran *malware* Trojan adalah yang paling pesat ke berbagai platform elektronik termasuk Android. Beberapa *malware* yang dapat ditemui pada perangkat Android bermacam-macam mulai dari BaseBridge, JIFake, KungFu, Fakedolphin, SNDApps, FakeInst, VDLoader dan masih banyak lagi.

2.2.1.1 Jenis Malware

Ada beberapa jenis malware yang perlu diketahui [6], yaitu:

1. Virus

   Virus merupakan jenis *malware* yang menyerang *file* eksekusi (.exe) yang akan menyerang dan menggandakan diri ketika *file* ext yang terinfeksi di jalankan. Virus komputer menyebar dengan cara menyisipkan program dirinya pada program atau dokumen dalam komputer. Jenis *malware* ini jarang ditemukan pada Android.

2. Worm

   Worm adalah salah satu jenis *malware* yang dapat menggandakan dirinya secara sendiri dalam sistem komputer. Sebuah worm dapat menggandakan dirinya dengan memanfaatkan jaringan (LAN/WAN/Internet)

tanpa perlu campur tangan dari *user* itu sendiri. Worm memanfaatkan celah keamanaan yang memang terbuka atau lebih dikenal dengan sebutan *vulnerability*.

3. Spyware

Spyware adalah jenis *malware* yang bertindak sebagai mata-mata untuk mengetahui kebiasaan pengguna komputer dan mengirimkan informasi tersebut ke pihak lain. Spyware biasanya digunakan oleh pihak pemasang iklan. Spyware adalah salah satu jenis *malware* yang banyak menyerang perangkat Android.

4. Trojan

Trojan atau bisa disebut juga sebagai Trojan Horse adalah program yang diam-diam masuk ke komputer kita, kemudian memfasilitasi program lain misalnya virus, sypware, adware. keylogger dan *malware* lainnya untuk masuk, merusak sistem, memungkinkan orang lain meremote komputer dan mencuri informasi seperti *password*. Trojan juga salah satu jenis *malware* yang sering ditemui di perangkat Android.

5. Adware

Adware adalah iklan yang dimasukan secara tersembunyi oleh pembuat program, biasanya pada program yang bersifat *freeware* untuk tujuan promosi atau iklan.

6. Keylogger

Keylogger adalah sebuah program yang dapat memantau penekanan tombol pada *keyboard*, sehingga orang lain dapat mengetahui *password* dan informasi apapun yang kita ketik.

7. Ransomware

Ransomware adalah bentuk *malware* yang pada dasarnya memegang sistem komputer sebagai tawanan sementara untuk menuntut tebusan. *Malware* ini membatasi akses pengguna ke komputer dengan mengenkripsi file pada *Hard Drive* ataupun mengunci sistem dan menampilkan pesan yang dimaksudkan untuk memaksa pengguna membayar pembuat *malware* (penyerang) agar mendapatkan kembali akses ke komputer mereka.

2.2.2 Reverse Engineering

Reverse Engineering atau dalam bahasa Indonesia Rekayasa Balik merupakan proses mencari dan menemukan prinsip kerja dari suatu produk teknologi yang ada pada sebuah sistem, perangkat atau objek, dengan menganalisis mendalam pada fungsi dan cara kerja dari sistem, perangkat atau objek yang diteliti [7]. Proses *reverse engineering* dapat dilakukan dengan cara [8].

1. Assembly

Assembly language merupakan bahasa pemrograman yang berada pada level rendah dari beberapa bahasa pemrograman yang dikenal selama ini. Bahasa *assembly* digunakan untuk sebuah mesin karena mesin tidak dapat

mengenal bahasa pemrograman tingkat tinggi seperti *java, basic, pascal,* dll.

2. Disassembly

Disassembly merupakan kebalikan dari proses *assembly*. Proses *disassembly* digunakan dalam teknik Reverse Engineering untuk menerjemahkan dari bahasa mesin ke bahasa yang mudah dimengerti manusia, yaitu bahasa *assembly*.

3. Debugging

Proses *debugging* adalah proses pengujian dari *software*. Pada analisa *malware debugging* digunakan untuk melakukan pengujian dari setiap proses inti yang ada didalam *malware*.

4. X86 Arsitektur

Dalam arsitektur x86 memiliki tiga komponen keras yaitu CPU, RAM, Input/Output (I/O). pada dasarnya pada internal dari kebanyakan arsitektur komputer modern yang termasuk juga x86 mengikuti arsitektur Von Neumann.

5. Instruction

Instruksi adalah konstruksi yang dibangun dari program *assembly*. Dalam *assembly* x86 instruksi terdiri dari *mnemonic* dan nol atau lebih *operands.*

6. Hashing

Hash merupakan identitas dari sebuah program seperti halnya sidik jari pada manusia. Proses *hash* dilakukan untuk verifikasi sebelum dan setelah proses analisa *malware*. Verifikasi tersebut dilakukan untuk mengetahui tidak adanya perubahan *hash* pada sample *malware* setelah dilakukan proses analisis.

7. String Analysis

String atau karakter dalam sebuah program seperti (.,A-) merupakan nilai yang akan dilakukan proses *load* oleh sampel *malware* ketika dieksekusi. Hal ini yang menjadikan dalam proses *reverse engineering* harus dilakukan *string* analisis untuk mendapatkan bukti kuat dari sampel *malware*.

8. MAER (Malware Analysis Environment and Requirment)

MAER adalah ruang lingkup yang menjadi laboratorium analisis malware. MAER merupakan salah satu penentu seorang analis malware mendapatkan infromasi yang akurat dan efisien dari analisa yang dilakukan.

9. Repository Malware

Repository malware merupakan tempat disimpannya sampel *malware* yang telah berhasil melakukan serangan ke dalam sistem komputer di manapun. *Repository malware* dibuat untuk memberikan sampel kepada seorang *malware* analis untuk melakukan analisa terhadap *malware* yang sudah berhasil melakukan serangan. *Repository Malware* adalah salah satu dari *malware source* yang dapat digunakan untuk kepentingan analisis. Selain menggunakan *repository*, sampel untuk analisis dapat pula berasal dari *honeypot* yang terpasang atau berdasarkan kasus yang dialami sendiri. Terdapat beberapa acuan untuk kepentingan sampel analisis *malware* yaitu : Virusshare, Contagio Malware Dump, Malshare, Malware.lu, MalwareBlacklist, MD Pro, Open Malware.

2.2.3 Remnux

Remnux adalah sistem operasi berbasis linux Ubuntu yang dibuat oleh Lenny Zeltser yang memiliki fungsi beragam. Yang membedakan Remnux dengan sistem operasi lainnya adalah Remnux menyediakan lebih banyak *tools* untuk melakukan analisis terhadap *malware* sehingga sistem operasi ini memiliki kelebihan dalam analisis *malware*, terutama dengan yang menggunakan metode *reverse engineering* [9]. Sebelumnya telah ada penelitian tentang analisis *malware* menggunakan Remnux dengan judul "*Analisis Malware Menggunakan Metode Reverse Engineering Pada Remnux"* yang di buat oleh Muhammad Rizal Zulfikar [1], pada penelitiannya Remnux di impor ke sebuah mesin virtual tidak dikonfigurasikan dengan Ubuntu. Sampel yang digunakan untuk analisa berformat EXE dan PDF, selain itu juga menganalisa *browser malware.* Didalam sistem operasi Remnux terdapat *tools* bernama Androguard yang dapat digunakan untuk menganalisa *file* berformat APK [10].

2.2.4 APK (Application Package File)

APK adalah format berkas yang digunakan untuk mendistribusikan dan memasang *software* dan *middleware* ke ponsel dengan sistem operasi Android. *File* APK berisi semua kode program (seperti *file* DEX), *Resources*, *Asset*, *Certificates*, dan *file Manifest* [11]. Pada umumnya analisa yang dilakukan pada *file* APK hanya dilakukan dengan mencari tahu *file Manifest* dan *file* berformat DEX.

3. ANALISIS DAN PERANCANGAN

3.1 Analisis

3.1.1 Gambaran Sistem Saat Ini

Saat ini alat untuk menganalisis sebuah aplikasi android dengan format APK membutuhkan 3 *tools* agar mendapatkan sebuah kode program dari sebuah aplikasi. Adapun gambaran sistem saat ini dapat dilihat pada Gambar 3.1.

Gambar 3. 1 Gambaran Sistem Saat Ini

3.1.2 Blok Diagram

Berikut ini adalah blok diagram sistem saat ini ketika menganalisis *malware.*

Gambar 3. 2 Blok Diagram Sistem

3.1.3 Cara Kerja Sistem

Sistem saat ini membutuhkan waktu yang panjang untuk memulai analisa terhadap sampel *malware*. Dimulai saat sampel di unduh dalam bentuk APK, maka agar dapat diekstrak harus mengubahnya menjadi file ZIP yang nantinya diekstrak menggunakan Winrar. Karena APK itu sendiri adalah kumpulan kode program berbentuk DEX(*Dalvik Executable*), folder *Resources*, *Asset*, *Certificates* dan *file Manifest.xml* yang dikompres, tetapi Winrar juga tidak dapat mengekstrak file berbentuk APK, maka dari itu diubah terlebih dahulu menjadi *file* berbentuk ZIP. Setelah selesai di ekstrak selanjutnya mengkonversi *file* DEX menjadi JAR (*Java Archive*) dengan *tools* Dex2jar. Langkah terakhir, *file* JAR

akan di *decompile* menggunakan *tools* JD-GUI sehingga dapat melihat semua *source code* Java yang akan diteliti dan dianalisa.

3.1.4 Analisis Kebutuhan Sistem

Berikut ini akan dijabarkan kebutuhan fungsional dan non fungsional.

3.1.4.1 Kebutuhan Fungsional

Kebutuhan fungsional yaitu sebagai berikut.

Table 3. 1 Kebutuhan Fungsional Sistem

No	Kebutuhan Fungsional
1	Memberikan memberikan informasi tentang kode programnya

3.1.4.2 Kebutuhan Non Fungsional

Kebutuhan non fungsional yaitu sebagai berikut.

Table 3. 2 Kebutuhan Non Fungsional Sistem

No	Kebutuhan Non Fungsional
1	Membutuhkan sebuah Laptop yang sudah di instalasikan 3 *tools* yaitu Winrar, Dex2jar dan JD-GUI.

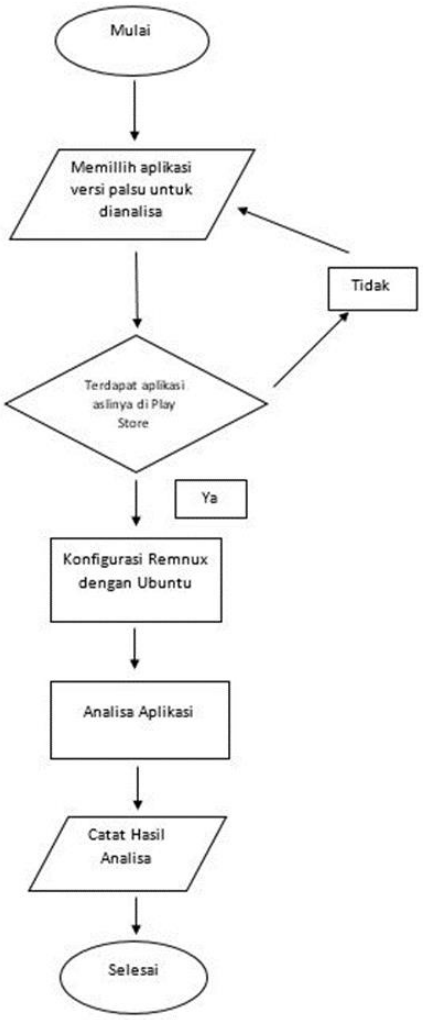
3.2 Perancangan

3.2.1 Gambaran Sistem Usulan

Sistem usulan hanya membutuhkan 1 *tools* yang ada pada sistem operasi Remnux versi 6 yang sebelumnya sudah di konfigurasikan pada sistem operasi Ubuntu versi 14.04. Adapun gambaran sistem usulan ini dapat dilihat pada Gambar 3.3.

Gambar 3. 3 Sistem Usulan

Adapun *flow chart* sistem usulan kali ini dapat dilihat pada Gambar 3.4. Penjelasan tentang cara kerja sistem dapat dilihat pada sub bab 3.2.3.



Gambar 3. 4 Flowchart Cara Kerja Sistem

3.2.2 Blok Diagram

Berikut ini adalah blok diagram sistem usulan saat menganalisis *malware.*

Gambar 3. 5 Blok Diagram Sistem Usulan

3.2.3 Cara Kerja

Sistem dalam proyek akhir ini menggunakan sebuah laptop yang menjalankan mesin virtual yang di dalamnya sudah terinstal sistem operasi Ubuntu 14.04 yang selanjutnya

mengkonfigurasikan Remnux dengan Ubuntu. Langkah berikutnya mengunduh sampel yang didapatkan dari sebuah situs web yang nantinya akan dibandingkan dengan versi aslinya yang didapatkan dari *Google Play Store* ataupun dari situs web yang menyediakan *file* APK asli. Selanjutnya membuka *tools* yang dibutuhkan untuk melakukan analisis dan memasukan *file* berkas APK yang akan dianalisa. Lamanya proses pembongkaran tergantung dari besarnya ukuran *file* APK yang akan dianalisa. Setelah proses pembongkaran selesai, langkah selanjutnya adalah menganalisa APK tersebut dengan memberikan suatu perintah, proses analisa dapat dilakukan dari *file manifest*.xml terlebih dahulu untuk memeriksa izin dan aktivitas atau memeriksa kode programnya langsung dengan melihat setiap *class* yang ada pada *file* APK tersebut.

3.2.4 Spesifikasi Sistem

Sistem ini membutuhkan perangkat keras dan beberapa perangkat lunak sebagai berikut.

3.2.4.1 Perangkat Keras

Adapun perangkat keras yang digunakan pada penelitian ini sebagai berikut.

Table 3. 3 Kebutuhan Perangkat Keras

No	Nama Hardware	Spesifikasi
1	Laptop	Core i3, RAM 6GB, HDD 500GB

3.2.4.2 Perangkat Lunak

Adapun perangkat keras yang digunakan pada penelitian ini sebagai berikut.

Table 3. 4 Kebutuhan Perangkat Keras

No	Nama Software	Spesifikasi
1	Sistem Operasi	Remnux ver.6 Ubuntu ver 14.04
2	Mesin Virtual	VirtualBox ver 5.1.26

4. PENGUJIAN

4.1 Investigate Mobile Malware dengan Androguard

Pengujian dilakukan dengan sampel *malware* yang dianalisis berupa aplikasi yang ada pada sistem operasi Android berformat APK yang didapatkan dari situs virusshare.com. Aplikasi yang akan dianalisa adalah *Angry_BirdsTransformers.apk*. Aplikasi ini adalah versi palsu dari aplikasi *game* bernama "Angry Birds Transformers" untuk Android. Lalu ada *Suivi_Conso.apk*, aplikasi ini adalah versi palsu dari aplikasi layanan pelacak rencana penggunaan perangkat android "SuiConFo". Yang terakhir adalah *Solitaire.apk*, aplikasi ini adalah versi palsu dari aplikasi *game* "Solitaire". Semua sampel tersebut akan dibandingkan dengan aplikasi asli yang didapatkan dari *Play Store Google* maupun situs web yang menyediakan aplikasi asli yang ada di *Play Store Google.*

4.2 Pengujian Analisa Sampel *Angry_BirdsTransformers.apk*

Pengujian yang dilakukan terhadap sampel pertama yaitu *Angry_BirdsTransformers.apk* terdapat beberapa perbedaan dengan aplikasi versi asli *AngryBirdsTransformers.apk* saat

melihat izin, nama paket dan aktivitas utama dari aplikasi.

Table 4. 1 Perbandingan *Angry_BirdsTransformers.apk* dengan *AngryBirdsTransformers.apk*

Angry_BirdsTransformers.apk	*AngryBirdsTransformers.apk*
Ada permintaan izin untuk membaca kontak, membaca penyimpanan eksternal, membaca, menerima, menulis dan mengirim SMS. Nama paket *com.elite* Aktivitas utama *com.elite.MainActivity*	Tidak ada permintaan izin untuk membaca kontak, membaca penyimpanan eksternal, membaca, menerima, menulis dan mengirim SMS. Nama paket *com.rovio.angrybirdstransformers* Aktivitas utama *com.rovio.angrybirdstransformers.AngryBirdsTransformersActivity*

Dipastikan bahwa aplikasi *Angry_BirdsTransformers.apk* mengandung kode program yang merugikan perangkat maupun pengguna dikarenakan tidak adanya nama developer pada nama peket dan aktivitas utama. Untuk mengetahui cara kerja *malware* yang tersembunyi di dalam aplikasi tersebut, dilakukan analisa terhadap kode program aplikasi yang ada pada setiap *class*.

Gambar 4. 1 *Class* yang dicurigai

Dari banyak *class* yang ada, terdapat 4 *class* yang dipastikan memiliki kode program *malware*. Pada *class* aktivitas utama yaitu *Lcom/elite/MainActivity* diketahui bahwa pada *class* tersebut memiliki kode program yang dapat menghapus seluruh direktori atau *file* yang ada pada penyimpanan eksternal (Gambar 4.2).

Gambar 4. 2 *Class Lcom/elite/MyServices$Async_sendSMS*

Selain itu pada *class Lcom/elite/MyServices$Async_sendSMS* ditemukan kode program yang dapat mengirim pesan SMS kesetiap nomor yang ada pada kontak telepon setiap 5 detik. Dampaknya adalah pulsa yang tiba-tiba habis padahal pengguna tidak mengirim pesan SMS (Gambar 4.3).

Gambar 4. 3 *Class Lcom/elite/MyServices$Async_sendSMS*

Pada *class Lcom/elite/MyServices* ditemukan kode program yang dapat mengunci layar perangkat dengan sebuah gambar satu layar penuh saat pengguna membuka aplikasi Facebook, Whatsapp, Google Hangouts dan Android SMS/MMS (Gambar 4.4).

Gambar 4. 4 Class Lcom/elite/MyServices

Yang terakhir aplikasi ini dapat menyembunyikan pesan SMS yang masuk dan selanjutnya membalas pesan tersebut ke pengirimnya dengan pesan yang mengancam tanpa sepengetahuan pengguna. Kode program tersebut ditemukan pada *class Lcom/elite/SMSReceiver* (Gambar 4.5).

Gambar 4. 5 *Class Lcom/elite/SMSReceiver*

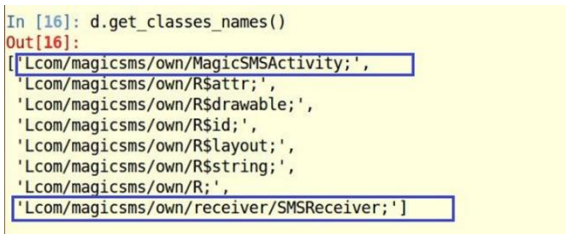
4.3 Pengujian Analisa Sampel Aplikasi *Suivi_Conso.apk*

Pengujian yang dilakukan terhadap sampel kedua yaitu *Suivi_Conso.apk* sama seperti pengujian sebelumnya, terdapat beberapa perbedaan dengan aplikasi versi asli *SuiConFo.apk* saat melihat izin, nama paket dan aktivitas utama dari aplikasi.

Table 4. 2 Perbandingan *Suivi_Conso.apk* dan *SuiConFo.apk*

Suivi_Conso.apk	*SuiConFo.apk*
Terdapat permintaan izin untuk mengirim dan menerima SMS Nama Paket *com.magicsms.own* Aktivitas Utama *com.magicsms.own. MagicSMSActivity*	Tidak ada permintaan izin untuk mengirim dan menerima SMS, tetapi ada permintaan untuk membaca SMS Nama paket *org.eo.suiviconso* Aktivitas utama *org.eo.suiviconso.Sui viConsoActivity*

Dipastikan bahwa aplikasi *Suivi_Conso.apk* mengandung kode program yang merugikan perangkat maupun pengguna dikarenakan tidak adanya nama developer pada nama peket dan aktivitas utama. Untuk mengetahui cara kerja *malware* yang tersembunyi didalam aplikasi tersebut, dilakukan analisa terhadap kode program aplikasi yang ada pada setiap *class*.

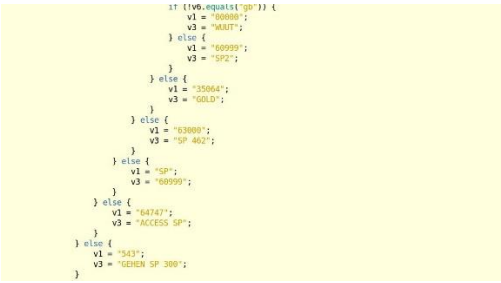


Gambar 4. 6 *Class* yang dicurigai

Di ketahui terdapat 2 *class* yang dicurigai berdasarkan hasil analisa izin, nama paket dan aktivitas utama dari aplikasi tersebut (Gambar 4.6).

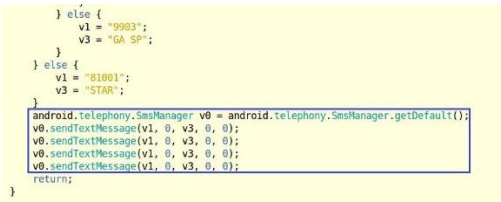
Gambar 4. 7 *Class Lcom/magicsms/own/MagicSMSActivity*

Pada *class* aktivitas utama yaitu *Lcom/magicsms/own/MagicSMSActivity* ditemukan terdapat kode perintah yang dapat menampilkan pesan *error* kepada pengguna, saat pengguna mulai menggunakan aplikasi tersebut (Gambar 4.7).



Gambar 4. 8 Kode Negara dan Pesan SMS yang akan dikirim

Lalu pada gambar 4.8 dapat diketahui bahwa aplikasi ini mencoba untuk mengirim pesan SMS ke beberapa negara dengan kode negara dan pesan SMS yang ada pada kode program yang selanjutnya akan dikirim sebanyak 4 pesan SMS (Gambar 4.9).



Gambar 4. 9 Kode Program untuk mengirim pesan SMS sebanyak 4 kali

Selanjutnya pada analisa kode program yang ada pada *class Lcom/magicsms/own/receiver/SMSReceiver* ditemukan bahwa juga dapat menyembunyikan pesan SMS yang masuk dan mengirimnya kembali seperti hasil analisa terhadap sampel pertama (Gambar 4.10).

Gambar 4. 10 *Class Lcom/magicsms/own/receiver/SMSReceiver*

4.4 Pengujian Analisa Sampel Aplikasi *Solitaire.apk*

Pengujian yang dilakukan terhadap sampel kedua yaitu *Solitaire.apk* sama seperti pengujian sebelumnya, terdapat beberapa perbedaan dengan aplikasi versi asli *SolitaireClassic.apk* saat melihat izin, nama paket dan aktivitas utama dari aplikasi.

Table 4. 3 Perbandingan *Solitaire.apk* dan *SolitaireClassic.apk*

Solitaire.apk	*SolitaireClassic.apk*
Terdapat izin untuk menerima *booting* selesai	Tidak terdapat izin untuk menerima *booting* selesai
Nama paket *com.tutusw.phonespeedup*	Nama paket *com.trumpgamestudio.game*
Aktivitas Utama *com.tutusw.phonespeedupIntroActivity.*	Aktivitas Utama *com.trumpgamestudio.game.AndroidLauncher*

Dipastikan bahwa aplikasi *Solitaire.apk* mengandung kode program yang merugikan perangkat maupun pengguna dikarenakan tidak adanya nama developer pada nama peket dan aktivitas utama. Untuk mengetahui cara kerja *malware* yang tersembunyi didalam aplikasi tersebut, dilakukan analisa terhadap kode program aplikasi yang ada pada setiap *class.*

Gambar 4. 11 *Class* yang dicurigai

Pada pengujian sebelumnya, *class* aktivitas utama pasti memiliki kode program yang

merugikan perangkat maupun pengguna, tetapi pada *Solitaire.apk* kode program tersebut tidak ditemukan di *class* aktivitas utama. Namun kode program yang merugikan dapat ditemukan pada *class Lcom/google/ssearch/Receiver* berdasarkan hasil analisa izin yang dibutuhkan aplikasi tersebut (Gambar 4.11).

Gambar 4. 12 *Class Lcom/google/ssearch/Receiver*

Pada *class Lcom/google/ssearch/Receiver* terdapat kode perintah untuk menjalankan *class Lcom/google/ssearch/SearchService* setelah menerima proses *booting* yang sudah selesai (Gambar 4.12).

Gambar 4. 13 *Class Lcom/google/ssearch/SearchService*

Lalu setelah melihat *class Lcom/google/ssearch/SearchService* (Gambar 4.13) terdapat perintah untuk memperbarui info tentang perangkat seperti nomor IMEI, model perangkat, operator yang digunakan, jaringan yang aktif aplikasi yang sedang berjalan dan lain-lain melalui perintah yang ada pada *class Lcom/google/ssearch/Utils$PhoneState.*

Gambar 4. 14 Fungsi *reportState()*

Yang selanjutnya pada gambar 4.14 masih di *class* yang sama, *malware* ini akan mencoba mengeksploitasi perangkat, jika tidak berhasi maka ia akan mencoba menggunakan biner *su* untuk menjadi *root*, dan mengekstrak sebuah APK Google Search palsu.

Gambar 4. 15 Fungsi *cpLegacyRes()*

Melalui aplikasi yang terinfeksi atau dari Google App palsu yang sudah terpasang dapat menerima perintah dari server jarak jauh yang dikendalikan oleh penyerang. Perintah ini dapat berupa menginstal atau menghapus paket apapun, memulai aplikasi atau membuka halaman web (Gambar 4.15).

Gambar 4. 16 Fungsi *doExecuteTask()*

5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian dan implementasi proyek akhir ini dapat diambil beberapa kesimpulan yaitu:

1. Pembangunan sistem berhasil dilakukan dengan mengkonfigurasikan Remnux versi 6 dengan Ubuntu versi 14.04.
2. Analisis yang dilakukan pada *file Angry_BirdsTransformers.apk* terbukti memiliki kode program yang berbahaya

dalam pengujiannya menggunakan *tools* Androguard. Seperti menghapus direktori dan isi *memory card*, menyembunyikan pesan yang masuk dan mengirim pesan secara bertubi-tubi. Analisis yang dilakukan pada *file Suivi_Conso.apk* terbukti memiliki kode program yang membuat perangkat yang terserang mengirimkan pesan ke beberapa nomor yang ada di beberapa negara Eropa dan Kanada tanpa sepengetahuan *user*. Analisis yang dilakukan pada *file Solitaire.apk* terbukti memiliki kode program jahat yang berjalan saat menerima proses *booting* yang sudah selesai yaitu mencuri data perangkat, seperti IMEI, tipe perangkat, versi OS dan lain-lain.

5.2 Saran

Beberapa saran yang dapat dilakukan untuk mengembangkan proyek akhir ini yaitu:

1. Dalam penggunaan Remnux sebagai sistem analisis dapat menggunakan SIFT Workstation juga.
2. Membuat aplikasi sistem analisis yang dapat mendeteksi malware android dengan metode reverse engineering tanpa melihat kode program Java terlebih dahulu.

Daftar Pustaka

[1] M. R. Zulfikar, "Analisis Malware Menggunakan Metode Reverse Engineering Pada Remnux," pp. 13-14, 2017.

[2] R. Novrianda, Y. N. Kunang dan P. Shaksono, "Analisis Forensik Malware Pada Platform Android," pp. 1-4, 2014.

[3] BBC, "Google reveals Play Music All Access subscription service," 15 Mei 2013. [Online]. Available: https://www.bbc.co.uk/news/technology-22542725. [Diakses 30 Juli 2018].

[4] Android, "Youtube," [Online]. Available: https://www.youtube.com/watch?v=1CVbQttKUIk. [Diakses 2 Agustus 2018].

[5] B. Setyadi, "Ini dia, 4 Jenis Malware Paling Gahar di Indonesia," PCplus Online, [Online]. Available: https://www.pcplus.co.id/2014/11/berita-teknologi/ini-dia-4-jenis-malware-paling-gahar-di-indonesia/. [Diakses 5 Agustus 2018].

[6] C. Teknisi, "Pengertian Malware dan Jenis-Jenis Malware," [Online]. Available: http://www.catatanteknisi.com/2011/12/pengertian-jenis-jenis-malware.html. [Diakses 1 Agustus 2018].

[7] "Rekayasa Balik," Wikipedia, [Online]. Available: https://id.wikipedia.org/wiki/Rekayasa_balik. [Diakses 24 Juli 2018].

[8] H. A. Nugroho dan Y. Prayudi, "PENGGUNAAN TEKNIK REVERSE ENGINEERING PADA MALWARE ANALYSIS UNTUK IDENTIFIKASI SERANGAN MALWARE," Universitas Islam Indonesia, Yogyakarta, 2014.

[9] L. Zelster dan D. Westcoot, "REMnux: A Linux Toolkit for Reverse-Engineering and Analyzing Malware," [Online]. Available: https://remnux.org/. [Diakses 29 November 2017].

[10] L. Zeltser, "REMnux Docs," [Online]. Available: https://remnux.org/docs/distro/use/. [Diakses 25 Juli 2018].

[11] Kumandroid, "Pengertian APK," [Online]. Available: http://www.kumandroid.com/2015/11/pengertian-apk.html. [Diakses 1 Agustus 2018].